Date⇒ 22-01-2021
Module⇒ Backend
Lecture By⇒ Akash Handa
Subject ⇒ Introduction/Basics Of Backend

What is Npm?

npm is the world's largest Software Library (Registry)

npm is also a software Package Manager and Installer

The World's Largest Software Registry (Library)

npm is the world's largest Software Registry.

The registry contains over 800,000 code packages.

Open-source developers use npm to share software.

Many organizations also use npm to manage private development.

Using npm is Free

npm is free to use.

You can download all npm public software packages without any registration or logon.

Installing npm

npm is installed with Node.js

This means that you have to install Node.js to get npm installed on your computer.

Download Node.js from the official Node.js web site: https://nodejs.org

Software Package Manager

The name npm (Node Package Manager) stems from when npm first was created as a package manager for Node.js.

All npm packages are defined in files called package.json.

The content of package.json must be written in JSON.

At least two fields must be present in the definition file: name and version.

Example

```
{
"name" : "foo",
"version" : "1.2.3",
"description" : "A package for fooing things",
"main" : "foo.js",
"keywords" : ["foo", "fool", "foolish"],
"author" : "John Doe",
"licence" : "ISC"
}
```

Managing Dependencies

npm can manage dependencies.

npm can (in one command line) install all the dependencies of a project.

Dependencies are also defined in package.json.

dependencies

Dependency/Local Dependency

> Max Dependency local

> They work wrt to particular folder

Dev Dependency

> They work wrt to particular folder

> All those that use wrt to only development

Need at the time of development

Testing

Global Dependency

> This installed in the system

> We have to install only once

> They have to generate or run the app

> Need Admin access to install

> Very few Dependency will go global

What Is Package.json

All npm packages contain a file, usually in the project root, called package.json - this file holds various metadata relevant to the project. This file is used to give information to npm that allows it to identify the project as well as handle the project's dependencies. It can also contain other metadata such as a project description, the version of the project in a particular distribution, license information, even configuration data - all of which can be vital to both npm and to the end users of the package. The package.json file is normally located at the root directory of a Node.js project.
Node.js itself is only aware of two fields in the package.json:

```
{
  "name" : "barebones",
  "version" : "0.0.0",
}
```

In Short Package.json Contain below Details
> Description of app
> packages
> scripts
> meta(author,version)

What is Node.js?

- Node.js is an open source server environment
- Node.js is free
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the server

What Can Node.js Do?

- Node.js can generate dynamic page content
- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data
- Node.js can add, delete, modify data in your database

What is a Node.js File?

- Node.js files contain tasks that will be executed on certain events
- A typical event is someone trying to access a port on the server
- Node.js files must be initiated on the server before having any effect
- Node.js files have extension ".js"

Getting Started With It

Once you have downloaded and installed Node.js on your computer, let's try to display "Hello World" in a web browser.

Create a Node.js file "Filename.js", and add the following code:

```
var http = require('http');

http.createServer(function (req, res) {

  res.writeHead(200, {'Content-Type': 'text/html'});

  res.end('Hello World!');

}).listen(8080);
```

Save the file on your computer: C:\Users*Your Name*\filename.js

The code tells the computer to write "Hello World!" if anyone (e.g. a web browser) tries to access your computer on port 8080.

What is a Module in Node.js?

Consider modules to be the same as JavaScript libraries.

A set of functions you want to include in your application.

Include Modules

To include a module, use the `require()` function with the name of the module:

Now your application has access to the HTTP module, and is able to create a server:

```
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.end('Hello World!');
```

```
  }).listen(8080);
```

Create Your Own Modules

You can create your own modules, and easily include them in your applications.

The following example creates a module that returns a date and time object:

Example

Create a module that returns the current date and time:

```
exports.myDateTime = function () {
    return Date();
  };
```

Include Your Own Module

Now you can include and use the module in any of your Node.js files.

Example

Use the module "myfirstmodule" in a Node.js file:

```
var http = require('http');
var dt = require('./myfirstmodule');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write("The date and time are currently: " + dt.myDateTime());
  res.end();
}).listen(8080);
```
